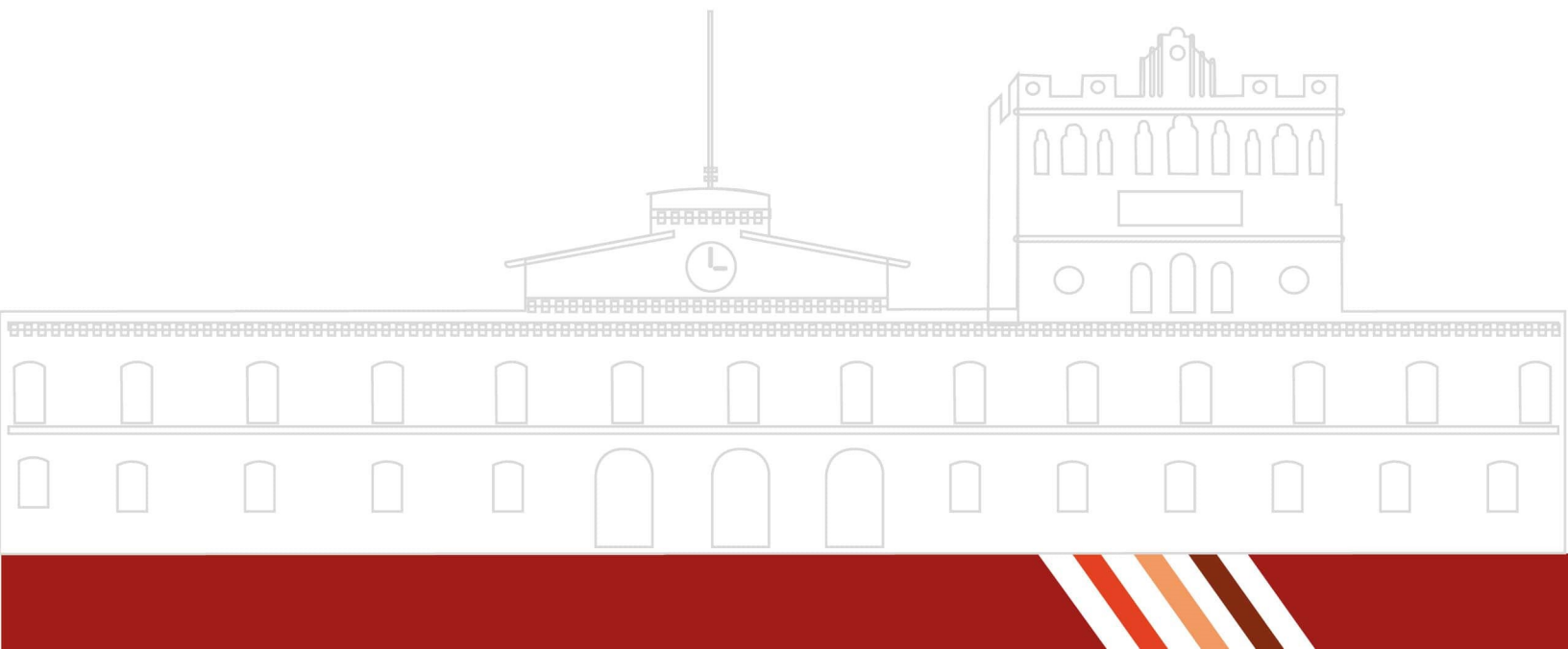


1.8 Práctica 2. AFD y AFND

NOMBRE DE LA PRÁCTICA: Opción 2

ALUMNO: Jorge Luis Ortega Pérez

Dr. Eduardo Cornejo-Velázquez



INSTRUCCIONES:

Resuelve el siguiente reto:

- <https://omegaup.com/arena/problem/Simulacion-de-AFD/>
- Sigue las instrucciones del reto y la condiciones que debe cumplir el programa. El entregable es el reporte de la práctica más el código del programa.

15320. Simulación de un Autómata Finito Determinista

Puntos	17.46	Límite de memoria	32 MiB
Límite de tiempo (caso)	1s	Límite de tiempo (total)	1m0s
Tamaño límite de entrada (bytes)	10 KiB		

DESCRIPCIÓN:

Un autómata finito determinista (AFD) es una 5-tupla $(Q, \Sigma, \delta, q_0, F)$, donde:

1. Q es un conjunto finito llamado **estados**,
2. Σ es un conjunto finito llamado **alfabeto**,
3. $\delta : Q \times \Sigma \rightarrow Q$ es la **función de transición**,
4. $q_0 \in Q$ es el **estado inicial**, y
5. $F \subseteq Q$ es el **conjunto de estados de aceptación**.

Sea $M = (Q, \Sigma, \delta, q_0, F)$ un AFD y sea $w = w_1 w_2 \dots w_n$ una cadena de símbolos donde $w_i \in \Sigma$. Entonces, M **acepta** w si existe una secuencia de estados r_0, r_1, \dots, r_n en Q con tres condiciones:

1. $r_0 = q_0$,
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, para $i = 0, \dots, n-1$, y
3. $r_n \in F$.

La condición 1 establece que la máquina comienza en el estado inicial. La condición 2 establece que la máquina cambia desde un estado hacia otro estado de acuerdo a la función de transición. La condición 3 establece que la máquina acepta la cadena de entrada si termina en un estado de aceptación. Entonces, M **reconoce el lenguaje** A si $A = \{w | M \text{ acepta } w\}$.

Escribir un programa para determinar si un conjunto de cadenas W pertenecen o no pertenecen al lenguaje A reconocido por un AFD M .

ENTRADA:

La primer línea de entrada contiene seis enteros N , S , D , q_0 , T y C , ($1 \leq N, S, C \leq 100$, $1 \leq D \leq 10^4$, $1 \leq q_0 \leq N$, $0 \leq T \leq N$), donde $N = |Q|$, $S = |\Sigma|$, $D = N \times S$ es el número de transiciones en el autómata, q_0 es el estado inicial, $T = |F|$, y C es la cantidad de cadenas a verificar si son aceptadas o no por el autómata M . Cada estado $q \in Q$ se identifica de manera implícita por un número entero con valor entre 1 y N .

La segunda línea contiene el alfabeto Σ , representado por una secuencia de S símbolos s_i separados por espacios, tal que cada símbolo s_i puede ser una letra, un dígito o cualquier carácter del código ASCII excepto por el espacio.

La tercer línea contiene el conjunto de estados de aceptación F , representado por una secuencia de T enteros t_i ($1 \leq t_i \leq N$) separados por espacios.


Las siguientes D líneas especifican las transiciones del autómata. Cada línea define una transición $\delta(I, X) = J$ por medio de un entero I , un carácter X y un entero J ($I, J \in Q$ y $X \in \Sigma$) separados por espacios, representando la transición desde el estado I hacia el estado J cuando el símbolo de la entrada sea X .

Finalmente, cada una de las siguientes C líneas contienen una cadena W , compuesta por símbolos que pertenecen al alfabeto Σ . La longitud de la cadena W está entre 0 y 100 caracteres.

SALIDA:

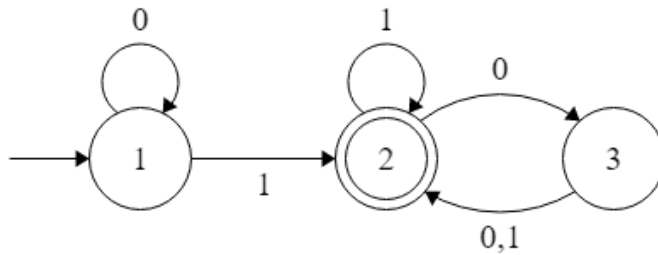
Para cada una de las C cadenas W se deberá imprimir el mensaje **ACEPTADA** si el autómata M acepta la cadena W , ó **RECHAZADA** en caso contrario.

EJEMPLO:

Entrada	Salida
3 2 6 1 1 3 	ACEPTADA
0 1	RECHAZADA
2	ACEPTADA
1 0 1	
1 1 2	
2 0 3	
2 1 2	
3 0 2	
3 1 2	
100	
0	
11	

DESCRIPCIÓN DEL EJEMPLO:

En el ejemplo el autómata tiene $N = 3$ estados, un alfabeto con $S = 2$ símbolos (segunda línea $\Sigma = \{0, 1\}$), se definen $D = 6$ transiciones (líneas 4 – 9), el estado inicial es $q_0 = 1$, solamente hay $T = 1$ estado de aceptación (tercer línea $F = \{2\}$) y se verifican $C = 3$ palabras W (líneas 10 – 12). La palabra 100 es aceptada, 0 es rechazada y 11 es aceptada. El autómata reconoce palabras que tienen una cantidad par de 0's después del último 1. La siguiente figura es una representación gráfica del autómata de ejemplo:



CODIGO DEL EJERCICIO:

```
include <iostream>
include <vector>
include <map>
include <set>
include <string>

using namespace std;

// Definición de la función de transición
map<pair<int, char>, int> delta;

// Conjunto de estados de aceptación
set<int> estados_aceptacion;

// Función que simula el AFD
bool simularAFD(int estado_inicial, const string cadena)
int estado_actual = estado_inicial;
for (char simbolo : cadena)
    auto transicion = make_pair(estado_actual, simbolo);
    if (delta.find(transicion) != delta.end())
        estado_actual = delta[transicion];
    else
        // No hay transición definida para este símbolo
        return false;
// Verificar si el estado actual es de aceptación
return estados_aceptacion.find(estado_actual) != estados_aceptacion.end();

int main()
// Definir los estados y el alfabeto
int num_estados;
cout << "Ingrese el número de estados: ";
```

```
cin >> num_estados;
```

```
int num_simbolos;  
cout << "Ingrese el número de símbolos en el alfabeto: ";  
cin >> num_simbolos;
```

```
vector<char> alfabeto(num_simbolos);  
cout << "Ingrese los símbolos del alfabeto: ";  
for (int i = 0; i < num_simbolos; ++i)  
cin >> alfabeto[i];
```

```
// Definir el estado inicial  
int estado_inicial;  
cout << "Ingrese el estado inicial: ";  
cin >> estado_inicial;
```

```
// Definir los estados de aceptación  
int num_estados_aceptacion;  
cout << "Ingrese el número de estados de aceptación: ";  
cin >> num_estados_aceptacion;
```

```
cout << "Ingrese los estados de aceptación: ";  
for (int i = 0; i < num_estados_aceptacion; ++i)  
int estado;  
cin >> estado;  
estados_aceptacion.insert(estado);
```

```
// Definir la función de transición  
int num_transiciones;  
cout << "Ingrese el número de transiciones: ";  
cin >> num_transiciones;
```

```
cout << "Ingrese las transiciones en el formato 'estado_actualsimboloestado_siguiente' : ";  
for (int i = 0; i < num_transiciones; ++i)  
int estado_actual, estado_siguiente;  
char simbolo;  
cin >> estado_actual >> simbolo >> estado_siguiente;  
delta[make_pair(estado_actual, simbolo)] = estado_siguiente;
```

```
// Leer la cadena de entrada  
string cadena;  
cout << "Ingrese la cadena a evaluar: ";  
cin >> cadena;
```

```
// Simular el AFD
if (simularAFD(estado_inicial, cadena))
    cout << "La cadena es aceptada por el AFD.";
else
    cout << "La cadena no es aceptada por el AFD.";

return 0;
```